

## COMP 202 - Week 4

- We will now examine some program statements that allow us to make decisions and repeat certain instructions multiple times
- This week we focus on:
  - the flow of control through a method
  - decision-making statements
  - operators for making complex decisions
  - Repetition statements

## Flow of Control

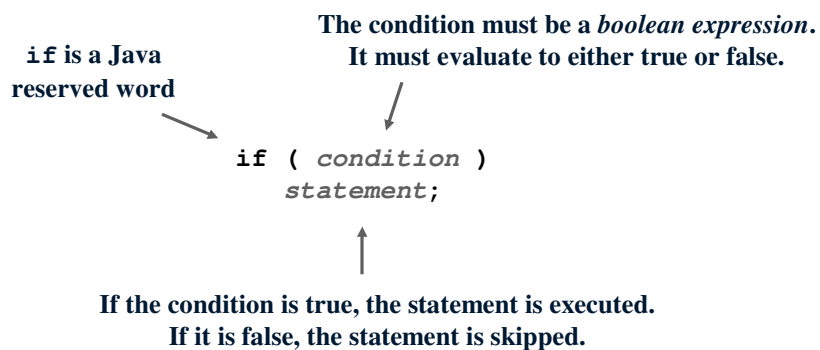
- Unless indicated otherwise, the order of statement execution through a method is linear: one after the other in the order they are written
- Some programming statements modify that order, allowing us to:
  - decide whether or not to execute a particular statement, or
  - perform a statement over and over repetitively
- The order of statement execution is called the *flow of control*

## Conditional Statements

- A *conditional statement* lets us choose which statement will be executed next
- Therefore they are sometimes called *selection statements*
- Conditional statements give us the power to make basic decisions
- Java's conditional statements are the *if statement*, the *if-else statement*, and the *switch statement*

## The if Statement

- The *if statement* has the following syntax:



## The if Statement

- An example of an if statement:

```
if (money > COST)
    money = money - COST;
System.out.println ("You have $" + money);
```

First, the condition is evaluated. The value of `money` is either greater than the value of `COST`, or it is not.

If the condition is true, the assignment statement is executed.  
If it is not, the assignment statement is skipped.

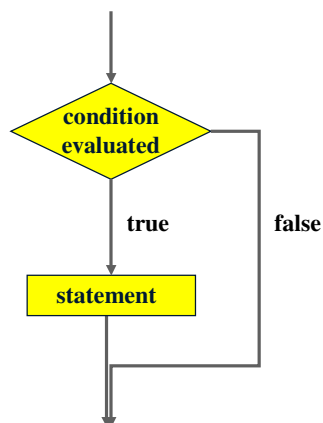
Either way, the call to `println` is executed next.

- See [BusRide.java](#)

COMP 202 - Week 4

5

## Logic of an if statement



COMP 202 - Week 4

6

## Boolean Expressions

- A condition often uses one of Java's *equality operators* or *relational operators*, which all return boolean results:

<code>==</code>	equal to
<code>!=</code>	not equal to
<code>&lt;</code>	less than
<code>&gt;</code>	greater than
<code>&lt;=</code>	less than or equal to
<code>&gt;=</code>	greater than or equal to

- Note the difference between the equality operator (`==`) and the assignment operator (`=`)

COMP 202 - Week 4

7

## The if-else Statement

- An *else clause* can be added to an if statement to make it an *if-else statement*:

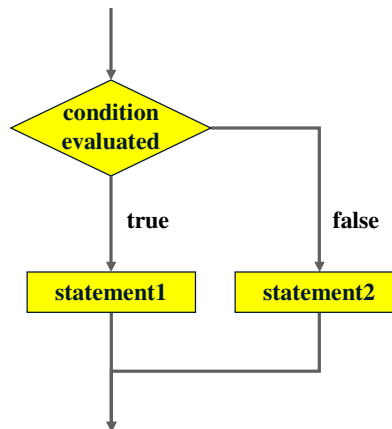
```
if ( condition )
    statement1;
else
    statement2;
```

- If the condition is true, statement1 is executed; if the condition is false, statement2 is executed
- One or the other will be executed, but not both
- See [Wages.java](#)

COMP 202 - Week 4

8

## Logic of an if-else statement



COMP 202 - Week 4

9

## Block Statements

- Several statements can be grouped together into a *block statement*
- A block is delimited by braces ( { . . . } )
- A block statement can be used wherever a statement is called for in the Java syntax
- For example, in an if-else statement, the if portion, or the else portion, or both, could be block statements
- See [Guessing.java](#)

COMP 202 - Week 4

10

## Nested if Statements

- The statement executed as a result of an if statement or else clause could be another if statement
- These are called *nested if statements*
- See [MinOfThree.java](#)
- An else clause is matched to the last unmatched if (no matter what the indentation implies)

COMP 202 - Week 4

11

## Comparing Characters

- We can use the relational operators on character data
- The results are based on the Unicode character set
- The following condition is true because the character '+' comes before the character 'J' in Unicode:

```
if ('+' < 'J')  
    System.out.println ("+ is less than J");
```

- The uppercase alphabet (A-Z) and the lowercase alphabet (a-z) both appear in alphabetical order in Unicode

COMP 202 - Week 4

12

## Comparing Strings

- Remember that a character string in Java is an object
- We cannot use the relational operators to compare strings
- The `equals` method can be called on a string to determine if two strings contain exactly the same characters in the same order
- The `String` class also contains a method called `compareTo` to determine if one string comes before another alphabetically (as determined by the Unicode character set)

COMP 202 - Week 4

13

## Comparing Floating Point Values

- We also have to be careful when comparing two floating point values (`float` or `double`) for equality
- You should rarely use the equality operator (`==`) when comparing two floats
- In many situations, you might consider two floating point numbers to be "close enough" even if they aren't exactly equal
- Therefore, to determine the equality of two floats, you may want to use the following technique:

```
if (Math.abs (f1 - f2) < 0.00001)
    System.out.println ("Essentially equal.");
```

COMP 202 - Week 4

14

## The switch Statement

- The *switch statement* provides another means to decide which statement to execute next
- The switch statement evaluates an expression, then attempts to match the result to one of several possible *cases*
- Each case contains a value and a list of statements
- The flow of control transfers to statement list associated with the first value that matches

COMP 202 - Week 4

15

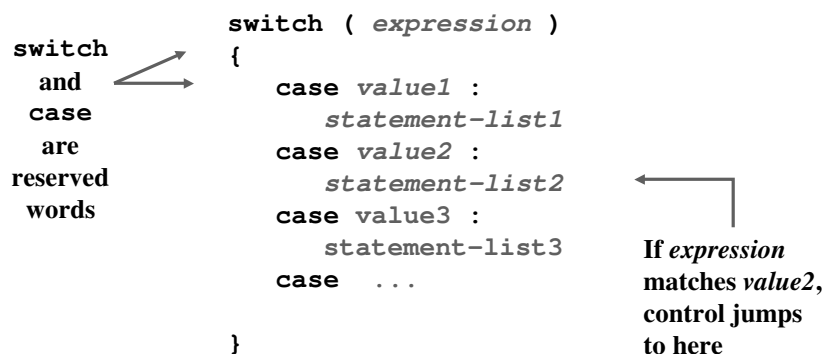
## The switch Statement

- The general syntax of a switch statement is:

switch  
and  
case  
are  
reserved  
words

```
switch ( expression )  
{  
    case value1 :  
        statement-list1  
    case value2 :  
        statement-list2  
    case value3 :  
        statement-list3  
    case ...  
}
```

If *expression*  
matches *value2*,  
control jumps  
to here



COMP 202 - Week 4

16

## The switch Statement

- Often a *break statement* is used as the last statement in each case's statement list
- A *break* statement causes control to transfer to the end of the switch statement
- If a break statement is not used, the flow of control will continue into the next case
- Sometimes this can be helpful, but usually we only want to execute the statements associated with one case

COMP 202 - Week 4

17

## The switch Statement

- A switch statement can have an optional *default case*
- The default case has no associated value and simply uses the reserved word `default`
- If the default case is present, control will transfer to it if no other case value matches
- Though the default case can be positioned anywhere in the switch, it is usually placed at the end
- If there is no default case, and no other value matches, control falls through to the statement after the switch

COMP 202 - Week 4

18

## The switch Statement

- The expression of a switch statement must result in an *integral data type*, like an integer or character; it cannot be a floating point value
- Note that the implicit boolean condition in a switch statement is equality - it tries to match the expression with a value
- You cannot perform relational checks with a switch statement
- See [AgeInLife.java](#)
- See [Drinks.java](#)

COMP 202 - Week 4

19

## Logical Operators

- Boolean expressions can also use the following *logical operators*:

!	Logical NOT
&&	Logical AND
	Logical OR

- They all take boolean operands and produce boolean results
- Logical NOT is a unary operator (it has one operand), but logical AND and logical OR are binary operators (they each have two operands)

COMP 202 - Week 4

20

## Logical NOT

- The *logical NOT* operation is also called *logical negation* or *logical complement*
- If some boolean condition **a** is true, then **!a** is false; if **a** is false, then **!a** is true
- Logical expressions can be shown using *truth tables*

<b>a</b>	<b>!a</b>
true	false
false	true

COMP 202 - Week 4

21

## Logical AND and Logical OR

- The *logical and* expression

**a && b**

is true if both **a** and **b** are true, and false otherwise

- The *logical or* expression

**a || b**

is true if **a** or **b** or both are true, and false otherwise

COMP 202 - Week 4

22

## Truth Tables

- A truth table shows the possible true/false combinations of the terms
- Since `&&` and `||` each have two operands, there are four possible combinations of true and false

a	b	a && b	a    b
true	true	true	true
true	false	false	true
false	true	false	true
false	false	false	false

COMP 202 - Week 4

23

## Logical Operators

- Conditions in selection statements and loops can use logical operators to form complex expressions

```
if (total < MAX && !found)
    System.out.println ("Processing...");
```

- Logical operators have precedence relationships between themselves and other operators

COMP 202 - Week 4

24

## Truth Tables

- Specific expressions can be evaluated using truth tables

<code>total &lt; MAX</code>	<code>found</code>	<code>!found</code>	<code>total &lt; MAX &amp;&amp; !found</code>
false	false	true	false
false	true	false	false
true	false	true	true
true	true	false	false

## The Conditional Operator

- Java has a *conditional operator* that evaluates a boolean condition that determines which of two other expressions is evaluated
- The result of the chosen expression is the result of the entire conditional operator
- Its syntax is:

`condition ? expression1 : expression2`

- If the *condition* is true, *expression1* is evaluated; if it is false, *expression2* is evaluated

## The Conditional Operator

- The conditional operator is similar to an if-else statement, except that it is an expression that returns a value
- For example:

```
larger = (num1 > num2) ? num1 : num2;
```

- If num1 is greater than num2, then num1 is assigned to larger; otherwise, num2 is assigned to larger
- The conditional operator is *ternary*, meaning that it requires three operands

COMP 202 - Week 4

27

## The Conditional Operator

- Another example:

```
System.out.println ("Your change is " + count +  
    (count == 1) ? "Dime" : "Dimes");
```

- If count equals 1, then "Dime" is printed
- If count is anything other than 1, then "Dimes" is printed
- See [Wages2.java](#)

COMP 202 - Week 4

28

## Repetition Statements

- **Repetition statements** allow us to execute a statement multiple times repetitively
- They are often simply referred to as *loops*
- Like conditional statements, they are controlled by boolean expressions
- Java has three kinds of repetition statements: the *while loop*, the *do loop*, and the *for loop*
- The programmer must choose the right kind of loop for the situation

COMP 202 - Week 4

29

## The while Statement

- The *while statement* has the following syntax:

**while** is a reserved word → **while** ( *condition* )  
*statement;*

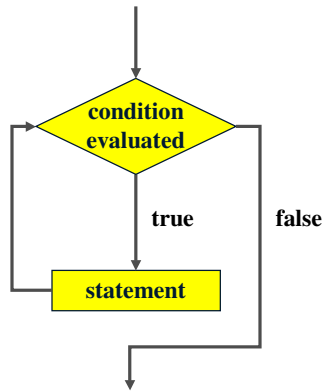
**If the condition is true, the statement is executed.  
Then the condition is evaluated again.**

**The statement is executed repetitively until the condition becomes false.**

COMP 202 - Week 4

30

## Logic of a while loop



COMP 202 - Week 4

31

## The while Statement

- Note that if the condition of a while statement is false initially, the statement is never executed
- Therefore, the body of a while loop will execute zero or more times
- See [Counter.java](#)
- See [BusSpeed.java](#)
- See [BusPercentage.java](#)

COMP 202 - Week 4

32

## Infinite Loops

- The body of a `while` loop must eventually make the condition false
- If not, it is an *infinite loop*, which will execute until the user interrupts the program
- See [Abyss.java](#)
- This is a common type of logical error
- You should always double check to ensure that your loops will terminate normally

COMP 202 - Week 4

33

## Nested Loops

- Similar to nested if statements, loops can be nested as well
- That is, the body of a loop could contain another loop
- Each time through the outer loop, the inner loop will go through its entire set of iterations
- See [PalindromeTester.java](#)

COMP 202 - Week 4

34

## The do Statement

- The *do statement* has the following syntax:

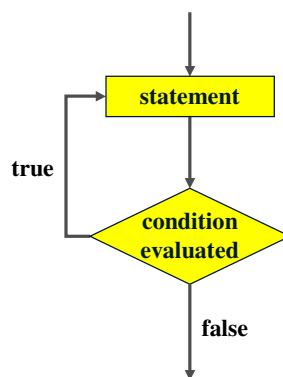
Uses both  
the **do** and  
**while**  
reserved  
words

```
do  
{  
    statement;  
}  
while ( condition );
```

The statement is executed once initially, then the condition is evaluated

The statement is repetitively executed until the condition becomes false

## Logic of a do loop



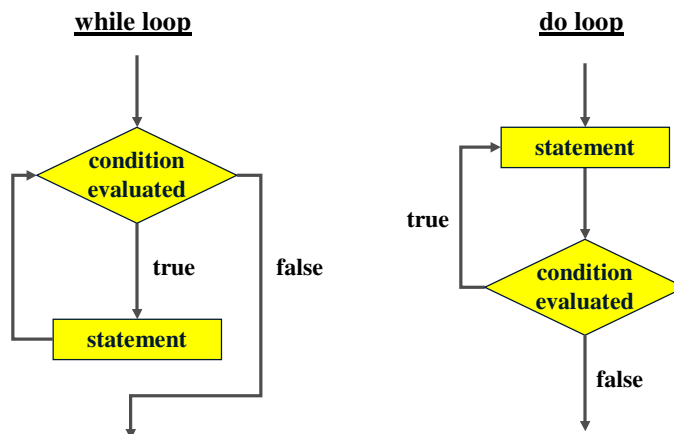
## The do Statement

- A do loop is similar to a while loop, except that the condition is evaluated after the body of the loop is executed
- Therefore the body of a do loop will execute at least one time
- See [Counter2.java](#)
- See [ReverseNumber.java](#)

COMP 202 - Week 4

37

## Comparing the while and do loops

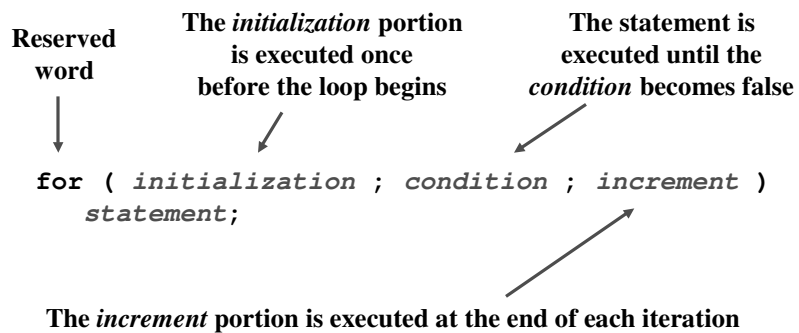


COMP 202 - Week 4

38

## The for Statement

- The *for* statement has the following syntax:

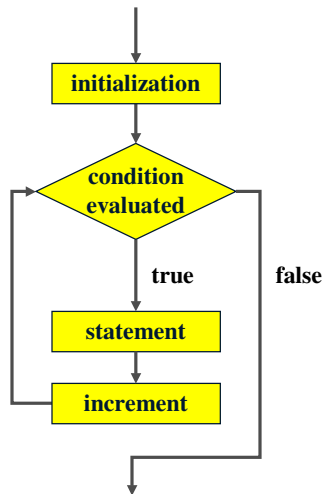


## The for Statement

- A *for* loop is equivalent to the following *while* loop structure:

```
initialization;
while ( condition )
{
    statement;
    increment;
}
```

## Logic of a for loop



COMP 202 - Week 4

41

## The for Statement

- Like a while loop, the condition of a for statement is tested prior to executing the loop body
- Therefore, the body of a for loop will execute zero or more times
- It is well suited for executing a specific number of times that can be determined in advance
- See [Counter3.java](#)
- See [Multiples.java](#)
- See [Stars.java](#)
- See [Christmas.java](#)

COMP 202 - Week 4

42

## **The for Statement**

- **Each expression in the header of a for loop is optional**
  - If the initialization is left out, no initialization is performed
  - If the condition is left out, it is always considered to be true, and therefore creates an infinite loop
  - If the increment is left out, no increment operation is performed
- **Both semi-colons are always required in the for loop header**